Context Uncertainty in Contextual Bandits with Applications to Recommender Systems

Hao Wang1,2*, Yifei Ma1, Hao Ding1, Yuyang Wang1
1AWS AI Labs 2Department of Computer Science, Rutgers University
hw488@cs.rutgers.edu, {yifeim,haodin,yuyawang}@amazon.com

Abstract

Recurrent neural networks have proven effective in modeling sequential user feedbacks for recommender systems. However, they usually focus solely on item relevance and fail to effectively explore diverse items for users, therefore harming the system performance in the long run. To address this problem, we propose a new type of recurrent neural networks, dubbed recurrent exploration networks (REN), to jointly perform representation learning and effective exploration in the latent space. REN tries to balance relevance and exploration while taking into account the uncertainty in the representations. Our theoretical analysis shows that REN can preserve the rate-optimal sublinear regret even when there exists uncertainty in the learned representations. Our empirical study demonstrates that REN can achieve satisfactory long-term rewards on both synthetic and real-world recommendation datasets, outperforming state-of-the-art models.

Introduction

Modeling and predicting sequential user feedbacks is a core problem in modern e-commerce recommender systems. In this regard, recurrent neural networks (RNN) have shown great promise since they can naturally handle sequential data (Hidasi et al. 2016; Quadran et al. 2017; Belletti, Chen, and Chi 2019; Ma et al. 20). While these RNN-based models can effectively learn representations in the latent space to achieve satisfactory immediate recommendation accuracy, they typically focus solely on relevance and fall short of effective exploration in the latent space, leading to poor performance in the long run. For example, a recommender system may keep recommending action movies to a user once it learns that she likes such movies. This may increase immediate rewards, but the lack of exploration in other movie genres can certainly be detrimental to long-term rewards.

So, how does one effectively explore diverse items for users while retaining the representation power offered by RNN-based recommenders. We note that the learned representations in the latent space are crucial for these models’ success. Therefore we propose recurrent exploration networks (REN) to explore diverse items in the latent space learned by RNN-based models. REN tries to balance relevance and exploration during recommendations using the learned representations.

One roadblock is that effective exploration relies heavily on well learned representations, which in turn require sufficient exploration; this is a chicken-and-egg problem. In a case where RNN learns unreasonable representations (e.g., all items have the same representations), exploration in the latent space is meaningless. To address this problem, we enable REN to take into account the uncertainty of the learned representations as well during recommendations. Essentially items whose representations have higher uncertainty can be explored more often. Such a model can be seen as a contextual bandit algorithm that is aware of the uncertainty for each context. Our contributions are as follows:

1. We propose REN as a new type of RNN to balance relevance and exploration during recommendation, yielding satisfactory long-term rewards.
2. Our theoretical analysis shows that there is an upper confidence bound related to uncertainty in learned representations. With such a bound implemented in the algorithm, REN can achieve the same rate-optimal sublinear regret.
3. Experiments of joint learning and exploration on both synthetic and real-world temporal datasets show that REN significantly improve long-term rewards over state-of-the-art RNN-based recommenders.

Related Work

ommender systems (Hidasi et al. 2016; Quadrana et al. 2017; Bai, Kolter, and Koltun 2018; Li et al. 2017; Liu et al. 2018; Wu et al. 2019; Ma et al. 2020), GRU4Rec (Hidasi et al. 2016) was first proposed to use gated recurrent units (GRU) (Cho et al. 2014), an RNN variant with gating mechanism, for recommendation. Since then, follow-up works such as hierarchical GRU (Quadrana et al. 2017), temporal convolutional networks (TCN) (Bai, Kolter, and Koltun 2018), and hierarchical RNN (HRNN) (Ma et al. 2020) have tried to achieve improvement in accuracy with the help of cross-session information (Quadrana et al. 2017), causal convolutions (Bai, Kolter, and Koltun 2018), as well as control signals (Ma et al. 2020). We note that our REN does not assume specific RNN architectures (e.g., GRU or TCN) and is therefore compatible with different RNN-based (or more generally DL-based) models, as shown in later sections.

Contextual Bandits. Contextual bandit algorithms such as LinUCB (Li et al. 2010) and its variants (Yue and Guestrin 2011; Agarwal et al. 2014; Li, Karatzoglou, and Gentile 2016; Kveton et al. 2017; Foster et al. 2018; Korda, Szorenyi, and Li 2016; Mahdik et al. 2020; Zhou, Li, and Gu 2019) have been proposed to tackle the exploitation-exploration trade-off in recommender systems and successfully improve upon context-free bandit algorithms (Auer 2002). Similar to (Auer 2002), theoretical analysis shows that LinUCB variants could achieve a rate-optimal regret bound (Chu et al. 2011). However, these methods either assume observed context (Zhou, Li, and Gu 2019) or are incompatible with neural networks (Li, Karatzoglou, and Gentile 2016; Yue and Guestrin 2011). In contrast, REN as a contextual bandit algorithm runs in the latent space and assumes user models based on RNN; therefore it is compatible with state-of-the-art RNN-based recommender systems.

Diversity-Inducing Models. Various works have focused on inducing diversity in recommender systems (Nguyen et al. 2014; Antikacioglu and Ravi 2017; Wilhelm et al. 2018; Bello et al. 2018). Usually such a system consists of a submodular function, which measures the diversity among items, and a relevance prediction model, which predicts relevance between users and items. Examples of submodular functions include the probabilistic coverage function (Hiranandani et al. 2019) and facility location diversity (FILD) (Tschiatschek, Djolonga, and Krause 2016), while relevance prediction models can be Gaussian processes (Vanchinathan et al. 2014), linear regression (Yue and Guestrin 2011), etc. These models typically focus on improving diversity among recommended items in a slate at the cost of accuracy. In contrast, REN’s goal is to optimize for long-term rewards through improving diversity between previous and recommended items. We include some slate generation in our real-data experiments for completeness.

Recurrent Exploration Networks
In this section we first describe the general notations and how RNN can be used for recommendation, briefly review determinantal point processes (DPP) as a diversity-inducing model as well as their connection to exploration in contextual bandits, and then introduce our proposed REN framework.

Notation and RNN-Based Recommender Systems

Notation. We consider the problem of sequential recommendations where the goal is to predict the item a user interacts with (e.g., click or purchase) at time $t$, denoted as $e_k$, given her previous interaction history $E_t = [e_{k-1}]_{k=1}^{t}$. Here $k_t$ is the index for the item at time $t$, $e_{k_t} \in \{0, 1\}^K$ is a one-hot vector indicating an item, and $K$ is the number of total items. We denote the item embedding (encoding) for $e_k$ as $x_k = f_e(e_k)$, where $f_e(\cdot)$ is the encoder as a part of the RNN. Correspondingly we have $X_t = [x_{k_t}]_{t=1}^{T}$. Strictly speaking, in an online setting where the model updates at every time step $t$, $x_k$ also changes over time; in Sec. we use $x_k$ as a shorthand for $x_{t,k}$ for simplicity. We use $||z||_\infty = \max_i |z^{(i)}|$ to denote the $L_\infty$ norm, where the superscript $(i)$ means the $i$-th entry of the vector $z$.

RNN-Based Recommender Systems. Given the interaction history $E_t$, the RNN generates the user embedding at time $t$ as $\theta_t = R([x_k]_{t=1}^{T})$, where $x_k = f_e(e_k) \in \mathcal{R}^d$, and $R(\cdot)$ is the recurrent part of the RNN. Assuming tied weights, the score for each candidate item is then computed as $p_{k,t} = x_k^\top \theta_t$. As the last step, the recommender system will recommend the items with the highest scores to the user. Note that the subscript $k$ indexes the items, and is equivalent to an ‘action’, usually denoted as $a$, in the context of bandit algorithms.

Determinantal Point Processes for Diversity and Exploration

Determinantal point processes (DPP) consider an item selection problem where each item is represented by a feature vector $x_i$. Diversity is achieved by picking a subset of items to cover the maximum volume spanned by the items, measured by the log-determinant of the corresponding kernel matrix, $\log(\det(I + \mathbf{X} \mathbf{X}^\top))$, where $I$ is included to prevent singularity. Intuitively, DPP penalizes colinearity, which is an indicator that the topics of one item are already covered by the other topics in the full set. The log-determinant of a kernel matrix is also a submodular function (Friedland and Gaubert 2013), which implies a $(1 - 1/e)$-optimal guarantees from greedy solutions. The greedy algorithm for DPP via the matrix determinant lemma is

$$\arg\max_k \log \det(I_d + X_t^\top X_t + x_k x_k^\top)$$

$$= \arg\max_k \log(1 + x_k^\top (I_d + X_t^\top X_t)^{-1} x_k)$$

$$= \arg\max_k \sqrt{x_k^\top (I_d + X_t^\top X_t)^{-1} x_k}.$$  

Interestingly, note that $\sqrt{x_k^\top (I_d + X_t^\top X_t)^{-1} x_k}$ has the same form as the confidence interval in LinUCB (Li et al. 2010), a commonly used contextual bandit algorithm to boost exploration and achieve long-term rewards, suggesting a connection between diversity and long-term rewards (Yue and Guestrin 2011). Intuitively, this makes sense in recommender systems since encouraging diversity relative to user history (as well as diversity in a slate of recommendations in our...
Algorithm 1: Recurrent Exploration Networks (REN)

1 Input: $\lambda_d, \lambda_u$, initialized REN model with the encoder, i.e., $R(\cdot)$ and $f_e(\cdot)$.
2 for $t = 1, 2, \ldots, T$ do
3     Obtain item embeddings from REN:
4         $\mu_k, \leftarrow f_e(e_k)$ for all $\tau \in \{1, 2, \ldots, t-1\}$.
5     Obtain current user embedding from REN:
6         $\theta_t \leftarrow R(D_t)$.
7     Compute $A_t \leftarrow I_d + \sum_{\tau \in \Psi_t} \mu_{k_{\tau}}^\top \mu_{k_{\tau}}$.
8     Obtain candidate items’ embeddings from REN:
9         $\mu_k \leftarrow f_e(e_k)$, where $k \in [K]$.
10    Obtain candidate items’ uncertainty estimates $\sigma_k$, where $k \in [K]$.
11    for $k \in [K]$ do
12         Obtain the score for item $k$ at time $t$:
13         $p_{k,t} \leftarrow \mu_k^\top \theta_t + \lambda_d \sqrt{\mu_k^\top A_t^{-1} \mu_k + \lambda_u \| \sigma_k \|_{\infty}}$.
14    end
15    Recommend item $k_t \leftarrow \text{argmax}_k p_{k,t}$ and collect user feedbacks.
16    Update the REN model $R(\cdot)$ and $f_e(\cdot)$ using collected user feedbacks.
17 end

experiments) naturally explores user interest previously unknown to the model, leading to much higher long-term rewards, as shown in Sec. 12.

Recurrent Exploration Networks

Exploration Term. Based on the intuition above, we can modify the user-item score $p_{k,t} = x_k^\top \theta_t$ to include a diversity (exploration) term, leading to the new score

$$p_{k,t} = x_k^\top \theta_t + \lambda_d \sqrt{x_k^\top (I_d + X_t^\top X_t)^{-1} x_k},$$

(4)

where the first term is the relevance score and the second term is the exploration score (measuring diversity between previous and recommended items). $\theta_t = R(X_t) = R([x_k]_{\tau=1}^{t-1})$ is RNN’s hidden states at time $t$ representing the user embedding. The hyperparameter $\lambda_d$ aims to balance two terms.

Uncertainty Term for Context Uncertainty. At first blush, given the user history the system using Eqn. 4 will recommend items that are (1) relevant to the user’s interest and (2) diverse from the user’s previous items. However, this only works when item embeddings $x_k$ are correctly learned. Unfortunately, the quality of learned item embeddings, in turn, relies heavily on the effectiveness of exploration, leading to a chicken-and-egg problem. To address this problem, one also needs to consider the uncertainty of the learned item embeddings. Assuming the item embedding $x_k \sim \mathcal{N}(\mu_k, \Sigma_k)$, where $\Sigma_k = \text{diag}(\sigma_k)$, we have the final score for REN:

$$p_{k,t} = \mu_k^\top \theta_t + \lambda_d \sqrt{\mu_k^\top (I_d + D_t^\top D_t)^{-1} \mu_k + \lambda_u \| \sigma_k \|_{\infty}},$$

(5)

where $\theta_t = R(D_t) = R([\mu_k]_{\tau=1}^{t-1})$ and $D_t = [\mu_k]_{\tau=1}^{t-1}$. The term $\sigma_k$ quantifies the uncertainty for each dimension of $x_k$, meaning that items whose embeddings REN is uncertain about are more likely to be recommended. Therefore with the third term, REN can naturally balance among relevance, diversity (relative to user history), and uncertainty during exploration.

Putting It All Together. Algorithm 1 shows the overview of REN. Note that the difference between REN and traditional RNN-based recommenders is only in the inference stage. During training (Line 16 of Algorithm 1), one can train REN only with the relevance term using models such as GRU4Rec and HRNN. In the experiments, we use uncertainty estimates $\text{diag}(\sigma_k) = 1/\sqrt{n_k} I_d$, where $n_k$ is item $k$’s total number of impressions (i.e., the number of times item $k$ has been recommended) for all users. The intuition is that: the more frequently item $k$ is recommended, the more frequently its embedding $x_k$ gets updated, the faster $\sigma_k$ decreases.\(^1\) Our preliminary experiments show that $1/\sqrt{n_k}$ does decrease at the rate of $O(1/\sqrt{t})$, meaning that the assumption in Lemma 4 is satisfied. From the Bayesian perspective, $1/\sqrt{n_k}$ may not accurately reflect the uncertainty of the learned $x_k$, which is a limitation of our model. In principle, one can learn $\sigma_k$ from data using the reparameterization trick (Kingma and Welling 2014) with a Gaussian prior on $x_k$ and examine whether $\sigma_k$ the assumption in Lemma 4; this would be interesting future work.

Linearity in REN. REN only needs a linear bandit model; REN’s output $x_k^\top \theta_t$ is linear w.r.t. $\theta_t$ and $x_k$. Note that NeuralUCB (Zhou, Li, and Gu 2019) is a powerful nonlinear extension of LinUCB, i.e., its output is nonlinear w.r.t. $\theta_t$ and $x_k$. Extending REN’s output from $x_k^\top \theta_t$ to a nonlinear function $f(x_k, \theta_t)$ as in NeuralUCB is also an interesting future work.\(^2\)

Beyond RNN. Note that our methods and theory go beyond RNN-based models and can be naturally extended to any latent factor models including transformers, MLPs, and matrix factorization. The key is the user embedding $\theta_t = R(X_t)$, which can be instantiated with an RNN, a transformer, or a matrix-factorization model.

Theoretical Analysis

With REN’s connection to contextual bandits, we can prove that with proper $\lambda_d$ and $\lambda_u$, Eqn. 5 is actually the upper confidence bound that leads to long-term rewards with a rate-optimal regret bound.

Reward Uncertainty versus Context Uncertainty. Note

\(^1\)There are some caveats in general. $\text{diag}(\sigma_k) \propto I_d$ assumes that all coordinates of $x$ shrink at the same rate. However, REN exploration mechanism associates $n_k$ with the total variance of the features of an item. This may not ensure all feature dimensions to be equally explored. See (Jun et al. 2019) for a different algorithm that analyzes the exploration of the low-rank feature space.

\(^2\)In other words, we did not fully explain why $x$ could be shared between non-linear RNN and the uncertainty bounds based on linear models. On the other hand, we did observe promising empirical results, which may encourage interested readers to dive deep into different theoretical analyses.
that unlike existing works which primarily consider the randomness from the reward, we take into consideration the uncertainty resulted from the context (content) (Mi et al. 2019; Xie, Wang, and Yeung 2016), i.e., context uncertainty. In CDL (Wang, Wang, and Yeung 2015; Wang, Shi, and Yeung 2016), it is shown that such content information is crucial in DL-based RecSys (Wang, Wang, and Yeung 2015; Wang, Shi, and Yeung 2016), and so is the associated uncertainty. More specifically, existing works assume deterministic x and only assume randomness in the reward, i.e., they assume that \( r = x^\top \theta + \epsilon \), and therefore r’s randomness is independent of x. The problem with this formulation is that they assume x is deterministic and therefore the model only has a point estimate of the item embedding x, but does not have uncertainty estimation for such x. We find that such uncertainty estimation is crucial for exploration; if the model is uncertain about x, it can then explore more on the corresponding item.

To facilitate analysis, we follow common practice (Auer 2002; Chu et al. 2011) to divide the procedure of REN into “BaseREN” (Algorithm 2) and “SupREN” (Algorithm 2) stages correspondingly. Essentially SupREN introduces \( S = \ln T \) levels of elimination (with s as an index) to filter out low-quality items and ensures that the assumption holds (see the Supplement for details of SupREN).

In this section, we first provide a high probability bound for BaseREN with uncertain embeddings (context), and derive an upper bound for the regret. As mentioned in Sec. , for the online setting where the model updates at every time step \( t \), \( x_t \) also changes over time. Therefore in this section we use \( x_{t,k}, \mu_{t,k}, \Sigma_{t,k}, \) and \( \sigma_{t,k} \) in place of \( x_k, \mu_k, \Sigma_k, \) and \( \sigma_k \) from Sec. to be rigorous.

**Assumption 1.** Assume there exists an optimal \( \theta^* \), with \|\theta^*\| \leq 1, and \( x^*_{t,k} \) such that \( E[r_{t,k}] = x^*_{t,k} \top \theta^* \). Further assume that there is an effective distribution \( N(\mu_{t,k}, \Sigma_{t,k}) \) such that \( x^*_{t,k} \sim N(\mu_{t,k}, \Sigma_{t,k}) \) where \( \Sigma_{t,k} = \text{diag}(\sigma_{t,k}^2) \). Thus, the true underlying context is unobservable, but we are aided with the knowledge that it is generated by a multivariate normal with known parameters.

**Upper Confidence Bound for Uncertain Embeddings**

For simplicity denote the item embedding (context) as \( x_{t,k} \), where \( t \) indexes the rounds (time steps) and \( k \) indexes the items. We define:

\[
s_{t,k} = \sqrt{\mu_{t,k} \top A_{t,k}^{-1} \mu_{t,k} \in \mathbb{R}_+}, \quad \Sigma_t = [\mu_{r,k}] \in \mathbb{R}^{|\Psi_t| \times d}, \quad y_t = [r_{t,k}] \in \mathbb{R}^{|\Psi_t|}, \quad A_t = I_d + D_t, \quad b_t = D_t \top y_t, \quad \hat{\mu}_{t,k} = \mu_{t,k} \top \theta_t = \mu_{t,k} \top A_{t,k}^{-1} b_t,
\]

where \( y_t \) is the collected user feedback. Lemma 1 below shows that with \( \lambda_2 = 1 + \alpha = 1 + \sqrt{\frac{2}{k} \ln \frac{TK}{\delta}} \) and \( \lambda_u = 4\sqrt{d} + 2\sqrt{\ln \frac{TK}{\delta}} \), Eqn. 5 is the upper confidence bound with high probability, meaning that Eqn. 5 upper bounds the true reward with high probability, which makes it a reasonable score for recommendations.

**Lemma 1 (Confidence Bound).** With probability at least \( 1 - 2\delta/T \), we have for all \( k \in [K] \) that

\[
|\hat{r}_{t,k} - x^*_{t,k} \top \theta^*| \leq (a + 1)s_{t,k}
\]

\[
+ (4\sqrt{d} + 2\sqrt{\ln \frac{TK}{\delta}}) \|\sigma_{t,k}\|_\infty,
\]

where \( \|\sigma_{t,k}\|_\infty = \max_i |\sigma_{t,k}^{(i)}| \) is the \( L_\infty \) norm.

The proof is in the Supplement. This upper confidence bound above provides important insight on why Eqn. 5 is reasonable as a final score to select items in Algorithm 1 as well as the choice of hyperparameters \( \lambda_2 \) and \( \lambda_u \).

**RNN to Estimate \( \theta_t \).** RNN uses RNN to approximate \( A_{t,k}^{-1} b_t \) (useful in the proof of Lemma 1) in Eqn. 6. Note that a linear RNN with tied weights and a single time step is equivalent to linear regression (LR); therefore RNN is a more general model to estimate \( \theta_t \). Compared to LR, RNN-based recommenders can naturally incorporate new user history by incrementally updating the hidden states (\( \theta_t \) in REN), without the need to solve a linear equation. Interestingly, one can also see RNN’s recurrent computation as a simulation (approximation) for solving equations via iterative updating.

**Regret Bound**

Lemma 1 above provides an estimate of the reward’s upper bound at time \( t \). Based on this estimate, one natural next step is to analyze the regret after all \( T \) rounds. Formally, we define the regret of the algorithm after \( T \) rounds as

\[
B(T) = \sum_{t=1}^{T} r_t k_t - \sum_{t=1}^{T} r_t k_t,
\]
where $k^*_t$ is the optimal item (action) $k$ at round $t$ that maximizes $E[r_t, k] = x_t^*k^* \theta^*$, and $k_t$ is the action chosen by the algorithm at round $t$. Similar to (Auer 2002), SupREN calls BaseREN as a sub-routine. In this subsection, we derive the regret bound for SupREN with uncertain item embeddings.

**Lemma 2.** With probability $1 - 2\delta S$, for any $t \in [T]$ and any $s \in [S]$, we have: (1) $|r_t, k - E[r_t, k]| \leq w_{t, k}$ for any $k \in [K]$; (2) $k^*_t \in A_s$; and (3) $E[r_t, k^*_t] - E[r_t, k] \leq 2^{(3-s)}$ for any $k \in A_s$.

**Lemma 3.** In BaseREN, we have: $(1 + \alpha) \sum_{t \in \Psi_{T+1}} s_{t, k_t} \leq 5 \cdot (1 + \alpha^2) \sqrt{d|\Psi_{T+1}|}$.

**Lemma 4.** Assuming $\|\sigma_{1,k}\|_\infty = 1$ and $\|\sigma_{1,k}\|_\infty \leq \frac{1}{\sqrt{T}}$ for any $k$ and $t$, then for any $k$, we have the upper bound: $\sum_{t \in \Psi_{T+1}} \|\sigma_{1,k}\|_\infty \leq \sqrt{|\Psi_{T+1}|}$.

Essentially Lemma 2 links the regret $B(T)$ to the width of the confidence bound $w_{t, k}$ (Line 9 of Algorithm 2 or the last two terms of Eqn. 5). Lemma 3 and Lemma 4 then connect $w_{t, k}$ to $\sqrt{|\Psi_{T+1}|} \leq \sqrt{T}$, which is sublinear in $T$; this is the key to achieve a sublinear regret bound. Note that $A_s$ is defined inside Algorithm 2 (SupREN) of the Supplement.

Interestingly, Lemma 4 states that the uncertainty only needs to decrease at the rate $\frac{1}{\sqrt{T}}$, which is consistent with our choice of $\text{diag}(\sigma_k) = 1/\sqrt{n_k} \mathbf{I}_d$ in Sec., where $n_k$ is item $k$’s total number of impressions for all users. As the last step, Lemma 5 and Theorem 1 below build on all lemmas above to derive the final sublinear regret bound.

**Lemma 5.** For all $s \in [S]$, $\left|\Psi^{(s)}_{T+1}\right| \leq 2^s \cdot (1 + \alpha^2) \sqrt{d|\Psi^{(s)}_{T+1}|} + 4\sqrt{dT} + 2\sqrt{T \ln \frac{TK}{\delta}}$.

**Theorem 1.** If SupREN is run with $\alpha = \sqrt{\frac{1}{2} \ln \frac{2TK}{\delta}}$, with probability at least $1 - \delta$, the regret of the algorithm is

$$B(T) \leq 2\sqrt{T} + 92 \cdot (1 + \ln \frac{2TK(2\ln T + 2)}{\delta}) \sqrt{T}d = O(\sqrt{Td \ln^3 \left(\frac{KT \ln(T)}{\delta}\right)})$$

All full proofs are in the Supplement. Theorem 1 shows that even with the uncertainty in the item embeddings (i.e., context uncertainty), our proposed REN can achieve the same rate-optimal sublinear regret bound.

**Experiments**

We evaluate REN on both synthetic and real-world datasets.

**Experiment Setup and Compared Methods**

**Joint Learning and Exploration Procedure in Temporal Data.** To effectively verify REN’s capability to boost long-term rewards, we adopt an online experiment setting where data is divided into different time intervals $[T_0, T_1], [T_1, T_2], \ldots, [T_{M-1}, T_M]$. RNN (including REN and its baselines) is then trained and evaluated in a rolling manner: (1) RNN is trained using data in $[T_0, T_1]$; (2) RNN is evaluated using data in $[T_1, T_2]$ and collects feedbacks (rewards) for its recommendations; (3) RNN uses newly collected feedbacks from $[T_1, T_2]$ to finetune the model; (4) Repeat the previous two steps using data from the next time interval. Note that different from traditional offline and one-step evaluation, corresponding to only Step (1) and (2), our setting performs joint learning and exploration in temporal data, and therefore is more realistic and closer to production systems.

**Long-Term Rewards.** Since the goal is to evaluate long-term rewards, we are mostly interested in the rewards during the last (few) time intervals. Conventional RNN-based recommenders do not perform exploration and are therefore much easier to saturate at a relatively low reward. In contrast, REN with its effective exploration can achieve nearly optimal rewards in the end.

**Compared Methods.** We compare REN variants with state-of-the-art RNN-based recommenders including GRU4Rec (Hidasi et al. 2016), TCN (Bai, Kolter, and Koltun 2018), HRNN (Ma et al. 2020). Since RNN can use any RNN-based recommenders as a base model, we evaluate three RNN variants in the experiments: REN-G, REN-T, and REN-H, which use GRU4Rec, TCN, and HRNN as base models, respectively. Additionally we also evaluate REN-1,2, an RNN variant without the third term of Eqn. 5, and REN-1,3, one without the second term of Eqn. 5, as an ablation study. Both REN-1,2 and REN-1,3 use GRU4Rec as the base model. As references we also include Oracle, which always achieves optimal rewards, and Random, which randomly recommends one item from the full set. For REN variants we choose $\lambda_d$ from $\{0.001, 0.005, 0.01, 0.05, 0.1\}$ and set $\lambda_u = \sqrt{10}\lambda_d$. Other hyperparameters in the RNN base models are kept the same for fair comparison (see the Supplement for more details on neural network architectures, hyperparameters, and their sensitivity analysis).

**Simulated Experiments**

**Datasets.** Following the setting described in Sec. 12, we start with three synthetic datasets, namely SYN-S, SYN-M, and SYN-L, which allow complete control on the simulated environments. We assume 8-dimensional latent vectors, which are unknown to the models, for each user and item, and use the inner product between user and item latent vectors as the reward. Specifically, for each latent user vector $\theta^*$, we randomly choose 3 entries to set to $1/\sqrt{3}$ and set the rest to 0, keeping $\|\theta^*\|_2 = 1$. We generate $C_8^3 = 28$ unique item latent vectors. Each item latent vector $x^*_j$ has 2 entries set to $1/\sqrt{2}$ and the other 6 entries set to 0 so that $\|x^*_j\|_2 = 1$.

We assume 15 users in our datasets. SYN-S contains exactly 28 items, while SYN-M repeats each unique item latent vector for 10 times, yielding 280 items in total. Similarly, SYN-L repeats 50 times, therefore yielding 1400 items in total. The purpose of allowing different items to have identical latent vectors is to investigate REN’s capability to explore in the compact latent space rather than the large item space. All users have a history length of 60.

**Simulated Environments.** With the generated latent vectors, the simulated environment runs as follows: At each time
As expected, conventional RNN-based recommenders saturate (first) and diversity (second) terms of Eqn. 5, saturates without the uncertainty term, REN-1,2 fails to effectively explore in exchange for long-term rewards.

**Results.** Fig. 1 shows the rewards over time for different methods. Results are averaged over 3 runs and we plot the rolling average with a window size of 100 to prevent clutter. As expected, conventional RNN-based recommenders saturate at around the 500-th time step, while all REN variants successfully achieve nearly optimal rewards in the end. One interesting observation is that REN variants obtain rewards lower than the "Random" baseline at the beginning, meaning that they are sacrificing immediate rewards to perform exploration in exchange for long-term rewards.

**Ablation Study.** Fig. 2 shows the rewards over time for REN-G (i.e., REN-1,2,3), REN-1,2, and REN-1,3 in SYN-S and SYN-L. We observe that REN-1,2, with only the relevance (first) and diversity (second) terms of Eqn. 5, saturates prematurely in SYN-S. On the other hand, the reward of REN-1,3, with only the relevance (first) and uncertainty (third) term, barely increases over time in SYN-L. In contrast, the full REN-G works in both SYN-S and SYN-L. This is because without the uncertainty term, REN-1,2 fails to effectively choose items with uncertain embeddings to explore. REN-1,3 ignores the diversity in the latent space and tends to explore items that have rarely been recommended; such exploration directly in the item space only works when the item number is small, e.g., in SYN-S.

**Hyperparameters.** For the base models GRU4Rec, TCN, and HRNN, we use identical network architectures and hyperparameters whenever possible. We set the number of hidden neurons to 32 for all models including REN variants.

**Real-World Experiments**

**MovieLens-1M.** We use MovieLens-1M (Harper and Konstan 2016) containing 3,900 movies and 6,040 users with an experiment setting similar to Sec. 12. Each user has 120 interactions, and we follow the joint learning and exploration procedure described in Sec. 12 to evaluate all methods (more details in the Supplement). All models recommend 10 items at each round for a chosen user, and the precision@10 is used as the reward. Fig. 3(left) shows the rewards over time averaged over all 6,040 users. As expected, REN variants with different base models are able to achieve higher long-term rewards compared to their non-REN counterparts.

**Trivago.** We also evaluate the proposed methods on Trivago4, a hotel recommendation dataset with 730,803 users, 926,457 items, and 910,683 interactions. We use a subset with 57,778 users, 387,348 items, and 108,713 interactions and slice the data into \( M = 48 \) one-hour time intervals for the online experiment (see the Supplement for details on data pre-processing). Different from MovieLens-1M, Trivago has impression data available: at each time step, besides which item is clicked by the user, we also know which 25 items are being shown to the user. Such information makes the online evaluation more realistic, as we now know the ground-truth feedback if an arbitrary subset of the 25 items is presented to the user. At each time step of the online experiments, all methods will choose 10 items from the 25 items to recommend the current user and collect the feedback for these 10 items as data for finetuning. We pretrain the model using all 25 items from the first 13 hours before starting the online evaluation. Fig. 3(middle) shows the mean reciprocal rank (MRR), the official metric used in the RecSys Challenge, for different

---

4More details are available at https://recsys.trivago.cloud/challenge/dataset/.
methods. As expected, the baseline RNN (e.g., GRU4Rec) suffers from a drastic drop in rewards because agents are allowed to recommend only 10 items, and they choose to focus only on relevance. This will inevitably ignores valuable items and harms the accuracy. In contrast, REN variants (e.g., REN-G) can effectively balance relevance and exploration for these 10 recommended items at each time step, achieving higher long-term rewards. Interestingly, we also observe that REN variants have better stability in performance compared to RNN baselines.

Netflix. Finally, we also use Netflix\(^5\) to evaluate how REN performs in the slate recommendation setting and without finetuning in each time step, i.e., skipping Step (3) in Sec. 12. We pretrain REN on data from half the users and evaluate on the other half. At each time step, REN generates 100 mutually diversified items for one slate following Eqn. 5, with \(p_k, \lambda\) updated after every item generation. Fig. 3(right) shows the recall@100 as the reward for different methods, demonstrating REN’s promising exploration ability when no finetuning is allowed (more results in the Supplement). Fig. 4(left) shows similar trends with recall@100 as the reward on the same holdout item set. This shows that the collected set contributes to building better user embedding models. Fig. 4(middle) shows that the additional exploration power comes without significant harms to the user’s immediate rewards on the exploration set, where the recommendations are served. In fact, we used a relatively large exploration coefficient, \(\lambda_d = \lambda_u = 0.0005\), which starts to affect recommendation results on the sixth position. By additional hyperparameter tuning, we realized that to achieve better rewards on the exploration set, we may choose smaller \(\lambda_d = 0.0007\) and \(\lambda_u = 0.0008\). Fig. 4(right) shows significantly higher recalls close to the oracle performance, where all of the users’ histories are known and used as inputs to predict the top-100 personalized recommendations.\(^6\) Note that, for fair presentation of the tuned results, we switched the exploration set and the holdout set and used a different test user group, consisting of 1543 users. We believe that the tuned results are generalizable with new users and items, but we also realize that the Netflix dataset still has a significant popularity bias and therefore we recommend using larger exploration coefficients with real online systems. The inference cost is 175 milliseconds to pick top-100 items from 8000 evaluation samples with 50 embedding dimensions, which is further improvable by selecting multiple items at a time in slate generation.

Conclusion

We propose the REN framework to balance relevance and exploration during recommendation. Our theoretical analysis and empirical results demonstrate the importance of considering uncertainty in the learned representations for effective exploration and improvement on long-term rewards.

\(^5\)https://www.kaggle.com/netflix-inc/netflix-prize-data

\(^6\)The gap between oracle and 100% recall lies in the model approximation errors.
Acknowledgements
The authors thank Tim Januschowski, Alex Smola, the AWS AI’s Personalize Team and ML Forecast Team, as well as the reviewers/SPC/AC for the constructive comments to improve the paper. We are also grateful for the RecoBandits package provided by Bharathan Blaji, Saurabh Gupta, and Jing Wang to facilitate the simulated environments. HW is partially supported by NSF Grant IIS-2127918.

References
Gupta, S.; Wang, H.; Lipton, Z.; and Wang, Y. 2021. Correcting exposure bias for link recommendation. In ICML.
Kingma, D. P.; and Welling, M. 2014. Auto-Encoding Variational Bayes. In ICLR.
for long range dependent user sequences. In WWW, 1782–1793.


